

QUESTION 10.



5 A company creates two new websites, Site X and Site Y, for selling bicycles.

Various programs are to be written to process the sales data.

These programs will use data about daily sales made from Site X (using variable SalesX), Site Y (using variable SalesY).

Data for the first 28 days is shown below.

	SalesDate	SalesX	SalesY
1	03/06/2015	0	1
2	04/06/2015	1	2
3	05/06/2015	3	8
4	06/06/2015	0	0
5	07/06/2015	4	6
6	08/06/2015	4	4
7	09/06/2015	5	9
8	10/06/2015	11	9
9	11/06/2015	4	1
...			
28	01/07/2015	14	8

(a) Name the data structure to be used in a program for SalesX.

.....[2]



(b) The programmer writes a program from the following pseudocode design.

```

x ← 0
FOR DayNumber ← 1 TO 7
  IF SalesX[DayNumber] + SalesY[DayNumber] >= 10
    THEN
      x ← x + 1
      OUTPUT SalesDate[DayNumber]
    ENDIF
  ENDFOR
OUTPUT x

```

(i) Trace the execution of this pseudocode by completing the trace table below.

x	DayNumber	OUTPUT
0		

[4]

(ii) Describe, in detail, what this algorithm does.

.....

.....

.....

.....[3]



- (c) The company wants a program to output the total monthly sales for one websites.

The programmer codes a function with the following function header:

```
FUNCTION MonthlyWebSiteSales(ThisMonth : INTEGER, ThisSite : CHAR)
                                RETURNS INTEGER
```

The function returns the total number of bicycles sold for the given month and website.

The function will use the following:

Identifier	Data type	Description
ThisMonth	INTEGER	Represents the month number e.g. 4 represents April
ThisSite	CHAR	Coded as: <ul style="list-style-type: none"> • X for website X • Y for Website Y

- (i) Give the number of parameters of this function.[1]
- (ii) Some of the following function calls may be invalid.

Mark each call with:

- a tick (✓), for a valid call
- a cross (✗), for an invalid call


For any function calls which are invalid, explain why.

Function call	Tick (✓) /cross (✗)	Explanation (if invalid)
MonthlyWebSiteSales(1, "Y")		
MonthlyWebSiteSales(11, 'X', 'Y')		
MonthlyWebSiteSales(12, 'X')		

[3]

- (d) The company decides to offer a discount on selected dates. A program is written to generate a text file, DISCOUNT_DATES, containing the dates on which a discount is offered.

The program creates a text file, DISCOUNT_DATES (with data as shown), for a number of consecutive dates.

03/06/2015 TRUE
04/06/2015 FALSE
05/06/2015 FALSE
06/06/2015 FALSE
07/06/2015 FALSE
08/06/2015 FALSE
09/06/2015 FALSE
10/06/2015 TRUE
11/06/2015 FALSE

01/07/2015 FALSE

Each date and discount indicator is separated by a single <Space> character.

The discount indicators are:

- FALSE – indicates a date on which no discount is offered
- TRUE – indicates a date on which a discount is offered

A programming language has the built-in function CONCAT defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )
                                                    RETURNS STRING
For example:   CONCAT("San", "Francisco") returns "SanFrancisco"
               CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.



The following incomplete pseudocode creates the text file DISCOUNT_DATES.

Complete the pseudocode.

```
OPENFILE "DISCOUNT_DATES" FOR .....  
INPUT .....  
WHILE NextDate <>"XXX"  
    INPUT Discount  
    ..... = CONCAT(NextDate, " ", Discount)  
    WRITEFILE "DISCOUNT_DATES", NextLine  
    INPUT NextDate  
    .....  
OUTPUT "File now created"  
CLOSEFILE
```

[4]



Question 5(e) continues on page 18.



(e) The `DISCOUNT_DATES` text file is successfully created.

The company now wants a program to:

- key in a date entered by the user
- search the text file for this date
- if found, output one of the following messages:
 - “No discount on this date”
 - “This is a discount date”
- if not found, output “Date not found”

(i) Add to the identifier table to show the variables you need for this new program.

Identifier	Data type	Description
<code>DISCOUNT_DATES</code>	<code>FILE</code>	Text file to be used

[3]

20

BLANK PAGE



QUESTION 11.



5 A company creates two new websites, Site X and Site Y, for selling bicycles.

Various programs are to be written to process the sales data.

These programs will use data about daily sales made from Site X (using variable SalesX), Site Y (using variable SalesY).

Data for the first 28 days is shown below.

	SalesDate	SalesX	SalesY
1	03/06/2015	0	1
2	04/06/2015	1	2
3	05/06/2015	3	8
4	06/06/2015	0	0
5	07/06/2015	4	6
6	08/06/2015	4	4
7	09/06/2015	5	9
8	10/06/2015	11	9
9	11/06/2015	4	1
...			
28	01/07/2015	14	8

(a) Name the data structure to be used in a program for SalesX.

.....[2]



(b) The programmer writes a program from the following pseudocode design.

```

x ← 0
FOR DayNumber ← 1 TO 7
    IF SalesX[DayNumber] + SalesY[DayNumber] >= 10
        THEN
            x ← x + 1
            OUTPUT SalesDate[DayNumber]
        ENDIF
    ENDFOR
OUTPUT x
    
```

(i) Trace the execution of this pseudocode by completing the trace table below.

x	DayNumber	OUTPUT
0		

[4]

(ii) Describe, in detail, what this algorithm does.

.....

.....

.....

.....[3]



- (c) The company wants a program to output the total monthly sales for one websites.

The programmer codes a function with the following function header:

```
FUNCTION MonthlyWebSiteSales(ThisMonth : INTEGER, ThisSite : CHAR)
                                RETURNS INTEGER
```

The function returns the total number of bicycles sold for the given month and website.

The function will use the following:

Identifier	Data type	Description
ThisMonth	INTEGER	Represents the month number e.g. 4 represents April
ThisSite	CHAR	Coded as: <ul style="list-style-type: none"> • X for website X • Y for Website Y

- (i) Give the number of parameters of this function.[1]
- (ii) Some of the following function calls may be invalid.

Mark each call with:

- a tick (✓), for a valid call
- a cross (✗), for an invalid call


For any function calls which are invalid, explain why.

Function call	Tick (✓) /cross (✗)	Explanation (if invalid)
MonthlyWebSiteSales(1, "Y")		
MonthlyWebSiteSales(11, 'X', 'Y')		
MonthlyWebSiteSales(12, 'X')		

[3]

- (d) The company decides to offer a discount on selected dates. A program is written to generate a text file, DISCOUNT_DATES, containing the dates on which a discount is offered.

The program creates a text file, DISCOUNT_DATES (with data as shown), for a number of consecutive dates.

03/06/2015 TRUE
04/06/2015 FALSE
05/06/2015 FALSE
06/06/2015 FALSE
07/06/2015 FALSE
08/06/2015 FALSE
09/06/2015 FALSE
10/06/2015 TRUE
11/06/2015 FALSE

01/07/2015 FALSE

Each date and discount indicator is separated by a single <Space> character.

The discount indicators are:

- FALSE – indicates a date on which no discount is offered
- TRUE – indicates a date on which a discount is offered

A programming language has the built-in function CONCAT defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )
                                                    RETURNS STRING
For example:   CONCAT("San", "Francisco") returns "SanFrancisco"
               CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.



The following incomplete pseudocode creates the text file DISCOUNT_DATES.

Complete the pseudocode.

```
OPENFILE "DISCOUNT_DATES" FOR .....  
INPUT .....  
WHILE NextDate <>"XXX"  
    INPUT Discount  
    ..... = CONCAT(NextDate, " ", Discount)  
    WRITEFILE "DISCOUNT_DATES", NextLine  
    INPUT NextDate  
    .....  
OUTPUT "File now created"  
CLOSEFILE
```

[4]



Question 5(e) continues on page 18.



(e) The `DISCOUNT_DATES` text file is successfully created.

The company now wants a program to:

- key in a date entered by the user
- search the text file for this date
- if found, output one of the following messages:
 - “No discount on this date”
 - “This is a discount date”
- if not found, output “Date not found”

(i) Add to the identifier table to show the variables you need for this new program.

Identifier	Data type	Description
<code>DISCOUNT_DATES</code>	<code>FILE</code>	Text file to be used

[3]

20

BLANK PAGE



QUESTION 12.

8 In this question you will need to use the given pseudocode built-in function:

ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
 returns the single character at position Position (counting from the start of the string with value 1) from the string ThisString.
 For example: ONECHAR("Barcelona", 3) returns 'r'.



(a) Give the value assigned to variable *y* by the following statement:

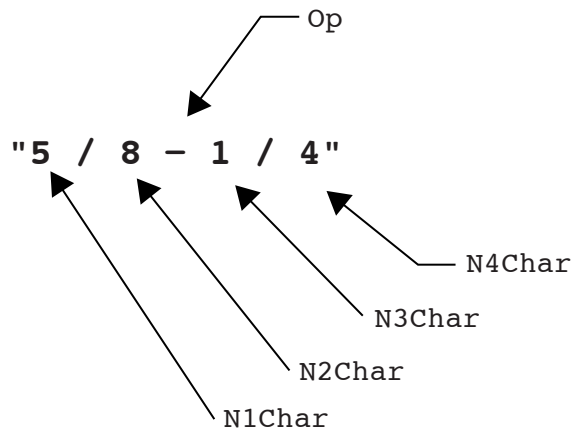
```
y ← ONECHAR("San Francisco", 6)           y ..... [1]
```

A program reads a string entered by the user. The string represents the addition or subtraction of two fractions. Each part of the fraction within the string is always a single digit only and the top digit is always less than the bottom digit.

Example strings are: "3/8+3/5" and "5/8-1/4"

The program steps are:

- the user enters the string
- the program isolates each digit and the operator
- the program computes the answer as either:
 - a fraction
 - a whole number followed by a fraction
 - a whole number
- the program displays the answer to the user



The identifier table shows the variables to be used to store the characters in the string as shown in the diagram.

Identifier	Data type	Description
FractionString	STRING	String input by user. For example: "5/8-1/4"
N1Char	CHAR	See diagram
N2Char	CHAR	See diagram
N3Char	CHAR	See diagram
N4Char	CHAR	See diagram
Op	CHAR	See diagram

QUESTION 13.



7 ASCII character codes are used to represent a single character.

Part of the code table is shown below.

ASCII code table (part)

Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

Some pseudocode statements follow which use these built-in functions:

`CHARACTERCOUNT(ThisString : STRING)` RETURNS INTEGER
 returns the number of characters in the string `ThisString`.
 For example: `CHARACTERCOUNT("South Africa")` returns 12.

`CHR(ThisInteger : INTEGER)` RETURNS CHAR
 returns the character with ASCII value `ThisInteger`.
 For example: `CHR(66)` returns 'B'.

`ASC(ThisCharacter : CHAR)` RETURNS INTEGER
 returns the ASCII value for character `ThisCharacter`.
 For example: `ASC('B')` returns 66.

(a) Give the values assigned to the variables A, B, C and D.

The & operator is used to concatenate two strings.

The expression could generate an error; if so, write ERROR.

<code>Num1 ← 5</code>
<code>A ← ASC('F') + Num1 + ASC('Z')</code>
<code>B ← CHR(89) & CHR(69) & CHR(83)</code>
<code>C ← CHARACTERCOUNT(B & "PLEASE")</code>
<code>D ← ASC(ONECHAR("CURRY SAUCE", 7))</code>

(i) A [1]

(ii) B [1]

(iii) C [1]

(iv) D [1]

- (ii) An experienced programmer suggests this pseudocode would be best designed as a function.

Complete the re-design of the pseudocode as follows:

The main program:

- the user enters `MyString`
- the function is called and the changed string is assigned to variable `ChangedString`

The function:

- has identifier `RemoveSpaces`
- has a single parameter
- will include the declaration for any local variables used by the function

```
// main program
INPUT MyString
ChangedString←RemoveSpaces (.....)
OUTPUT ChangedString

// function definition
FUNCTION RemoveSpaces (.....) RETURNS .....

.....

.....

.....

.....

j ← CHARACTERCOUNT (InputString)
FOR i ← 1 TO j
    NextChar ← ONECHAR (InputString, i)
    IF NextChar <> " "
        THEN
            // the & character joins together two strings
            NewString ← NewString & NextChar
        ENDIF
    ENDFOR
ENDFUNCTION
```

[7]

QUESTION 14.



6 A string-handling function has been developed. The pseudocode for this function is given below.

For the built-in functions list, refer to the **Appendix** on page 18.

```

FUNCTION SSM(String1, String2 : STRING) RETURNS INTEGER
    DECLARE n, f, x, y : INTEGER

    n ← 0
    f ← 0

    REPEAT
        n ← n + 1
        x ← n
        y ← 1
        WHILE MID(String1, x, 1) = MID(String2, y, 1)

            IF y = LENGTH(String2)
                THEN
                    f ← n
                ELSE
                    x ← x + 1
                    y ← y + 1
            ENDIF

        ENDWHILE

    UNTIL (n = LENGTH(String1)) OR (f <> 0)

    RETURN f

ENDFUNCTION
    
```

(a) Complete the trace table below by performing a dry run of the function when it is called as follows:

SSM("RETRACE", "RAC")

n	f	x	y	MID(String1, x, 1)	MID(String2, y, 1)
0	0				



(b) (i) Describe the purpose of function `SSM`.

.....
.....
.....
.....[2]

(ii) One of the possible return values from function `SSM` has a special meaning.

State the value and its meaning.

Value

Meaning

[2]

(iii) There is a problem with the logic of the pseudocode. This could generate a run-time error.

Describe the problem.

.....
.....
.....
.....[2]



Appendix

Built-in functions

In each function below, if the function call is not properly formed, the function returns an error.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`

returns the string of length `y` starting at position `x` from `ThisString`

Example: `MID ("ABCDEFGH", 2, 3)` will return string `"BCD"`

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`

returns the leftmost `x` characters from `ThisString`

Example: `LEFT ("ABCDEFGH", 3)` will return string `"ABC"`

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`

returns the rightmost `x` characters from `ThisString`

Example: `RIGHT ("ABCDEFGH", 3)` will return string `"FGH"`

`ASC(ThisChar : CHAR) RETURNS INTEGER`

returns the ASCII value of character `ThisChar`

Example: `ASC ('w')` will return `87`

`LENGTH(ThisString : STRING) RETURNS INTEGER`

returns the integer value representing the length of string `ThisString`

Example: `LENGTH ("Happy Days")` will return `10`

String operator

`&` operator

concatenates (joins) two strings

Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`



QUESTION 15.



6 A string-handling function has been developed.

For the built-in functions list, refer to the **Appendix** on the last page.

The pseudocode for this function is shown below.

```
FUNCTION SF(ThisString : STRING) RETURNS STRING
  DECLARE x          : CHAR
  DECLARE NewString  : STRING
  DECLARE Flag       : BOOLEAN
  DECLARE m, n       : INTEGER

  Flag ← TRUE
  NewString ← ""
  m ← LENGTH(ThisString)

  FOR n ← 1 TO m

    IF Flag = TRUE
      THEN
        x ← UCASE(MID(ThisString, n, 1))
        Flag ← FALSE
      ELSE
        x ← LCASE(MID(ThisString, n, 1))
    ENDIF

    NewString ← NewString & x

    IF x = " "
      THEN
        Flag ← TRUE
    ENDIF

  ENDFOR

  RETURN NewString
ENDFUNCTION
```

(a) (i) Complete the trace table below by performing a dry run of the function when it is called as follows:

SF("big BEN")

n	x	Flag	m	NewString



(ii) Describe the purpose of function SF.

.....
.....
.....
.....[2]

(b) Test data must be designed for the function SF.

(i) State what happens when the function is called with an empty string.

.....
.....[1]

(ii) The function should be thoroughly tested.

Give **three** examples of non-empty strings that may be used.

In each case explain why the test string has been chosen.

String

Explanation

.....

String

Explanation

.....

String

Explanation

.....

[3]

QUESTION 16.

10



5 Study the following pseudocode statements.

```
CONST Pi = 3.1          : REAL

DECLARE Triangle, Base, Height, Radius, Cone : REAL

DECLARE a, b, c, Answer2 : INTEGER

DECLARE Answer1          : BOOLEAN

Base ← 2.6

Height ← 10

Triangle ← (Base * Height) / 2

Radius ← 1

Height ← 2

Cone ← 2 * Pi * Radius * (Radius + Height)

a ← 13

b ← 7

c ← 3

Answer1 ← NOT((a + b + c) > 28)

Total ← 34

Total ← Total - 2

Answer2 ← a + c * c
```

Give the final value assigned to each variable.

- | | |
|---------------------|-----|
| (i) Triangle | [1] |
| (ii) Cone | [1] |
| (iii) Answer1 | [1] |
| (iv) Total | [1] |
| (v) Answer2 | [1] |



Appendix

Built-in functions (pseudocode)

`ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR`

returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.

For example: `ONECHAR("New York", 5)` returns 'Y'

`CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER`

returns the number of characters in `ThisString`.

For example: `CHARACTERCOUNT("New York")` returns 8

`SUBSTR(ThisString : STRING, Value1 : INTEGER, Value2 : INTEGER) RETURNS STRING`

returns a sub-string from within `ThisString`.

`Value1` is the start index position (counting from the left, starting with 1).

`Value2` is the final index position.

For example: `SUBSTR("art nouveau", 5, 11)` returns "nouveau"

`TONUM(ThisString : STRING) RETURNS INTEGER or REAL`

returns the integer or real equivalent of the string `ThisString`.

For example: `TONUM("502")` returns the integer 502

`TONUM("56.36")` returns the real number 56.36

`ASC(ThisCharacter : CHAR) RETURNS INTEGER`

returns an integer which is the ASCII character code for the character `ThisCharacter`.

For example: `ASC('A')` returns integer 65

QUESTION 17.



6 Study the sequence of pseudocode statements.

```
CONST a = 3.2 : REAL
DECLARE x, y, z, Answer1, Answer2, Answer3 : REAL
DECLARE p, q : BOOLEAN
x ← 3
x ← x + 7
y ← 6
Answer1 ← 2 * (a + y)
z ← 6
Answer2 ← y ^ 2 + 5
p ← TRUE
q ← NOT (NOT (p))
Answer3 ← y + a * 2
```

Give the final value assigned to each variable.

- (i) x [1]
- (ii) Answer1 [1]
- (iii) Answer2 [1]
- (iv) q [1]
- (v) Answer3 [1]



Appendix

Built-in functions (pseudocode)

ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR

returns the single character at position Position (counting from the start of the string with value 1) from the string ThisString.

For example: ONECHAR("New York", 5) returns 'Y'

CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER

returns the number of characters in ThisString.

For example: CHARACTERCOUNT("New York") returns 8

SUBSTR(ThisString : STRING, Value1 : INTEGER, Value2 : INTEGER) RETURNS STRING

returns a sub-string from within ThisString.

Value1 is the start index position (counting from the left, starting with 1).

Value2 is the final index position.

For example: SUBSTR("art nouveau", 5, 11) returns "nouveau"

TONUM(ThisString : STRING) RETURNS INTEGER or REAL

returns the integer or real equivalent of the string ThisString.

For example: TONUM("502") returns the integer 502

TONUM("56.36") returns the real number 56.36

ASC(ThisCharacter : CHAR) RETURNS INTEGER

returns an integer which is the ASCII character code for the character ThisCharacter.

For example: ASC('A') returns integer 65



`CHR(Value : INTEGER) RETURNS CHAR`

returns the character that ASCII code number `Value` represents.

For example: `CHR(65)` returns 'A'

`RND() RETURNS REAL`

returns a random number in the range 0 to 0.99999

For example: `RND()` returns 0.67351

`INT(ThisNumber : REAL) RETURNS INTEGER`

returns the integer part of `ThisNumber`.

For example: `INT(12.79)` returns 12

Errors

For any function, if the program calls the function incorrectly, the function returns an error.

Concatenation operator

`&` operator – Concatenates two expressions of `STRING` or `CHAR` data type.

For example: `"South" & " " & "Pole"` produces `"South Pole"`

`'B' & "000654"` produces `"B000654"`

QUESTION 18.



3 A string conversion function, `StringClean`, is to be written.

This function will form a new string, `OutString`, from a given string, `InString`, by:

- removing all non-alphabetic characters
- converting all alphabetic characters to lower case.

For example:

```
InString = "Good Morning, Dave"  
OutString = "goodmorningdave"
```

The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode using relevant built-in functions.

For the built-in functions list, refer to the **Appendix** on page 14.

```
FUNCTION StringClean(.....) RETURNS .....
```

```
    DECLARE NextChar : .....
```

```
    DECLARE ..... : STRING
```

```
    ..... //initialise the return string
```

```
    //loop through InString to produce OutString
```

```
    FOR n ← 1 TO ..... //from first to last
```

```
        NextChar ← ..... //get next character and
```

```
        NextChar ← ..... //convert to lower case
```

```
        IF ..... //check if alphabetic
```

```
            THEN
```

```
                ..... //add to OutString
```

```
            ENDIF
```

```
        ENDFOR
```

```
    .....//return value
```

```
ENDFUNCTION
```

QUESTION 19.



3 A string conversion function, `ExCamel`, needs to be written.

This function forms a return string, `OutString`, from a given string, `InString`, by:

- 1 separating the original words (a word is assumed to start with a capital letter)
- 2 converting all characters to lower case.

The following shows a pair of example values for the string values `InString` and `OutString`.

```
InString : "MyUserInput"  
OutString : "my user input"
```

You may assume that `InString` always starts with a capital letter.

The following is a first attempt at writing the pseudocode for this function.

Complete the **pseudocode** using appropriate built-in functions.

For the built-in functions list, refer to the **Appendix** on page 13.

```
FUNCTION ExCamel (.....) RETURNS .....  
  
    DECLARE NextChar : .....  
  
    DECLARE ..... : STRING  
  
    DECLARE n: INTEGER  
  
    ..... // initialise the return string  
  
    // loop through InString to produce OutString  
  
    FOR n ← 1 TO ..... // from first to last  
        NextChar ← ..... // get next character  
        IF ..... // check if upper case  
            THEN  
                IF n > 1 // if not first character  
                    THEN  
                        ..... // add space to OutString  
                    ENDIF  
                ..... // make NextChar lower case  
            ENDIF  
            ..... // add NextChar to OutString  
        ENDFOR  
  
        ..... // return value  
  
ENDFUNCTION
```


- 4 Numeric formatting converts a numeric value to a string in order to present it in a specific way.

In a generic high-level language, formatting is implemented using a mask system. In this system, each character of the mask corresponds to one character of the formatted string.

Mask characters have the following meaning:

Mask character	Meaning
#	Character must be a digit or a space
0	Character must be a digit

Any other mask characters are taken as literal values and are included in the formatted string.

- (a) Using the mask "###00.00", complete the following table. Use to represent a space. The first value has been done for you.

Value	Formatted string
1327.5	" <input type="checkbox"/> 1327.50"
1234	
7.456	

[2]

- (b) For each row in the following table, define the mask required to produce the formatted output from the given value. represents a space.

Value	Required output	Mask
1234.00	"1,234.00"	
3445.66	"£3,445.66"	
10345.56	"\$ <input type="checkbox"/> <input type="checkbox"/> 10,345"	

[3]

(b) Name **three** features of a typical IDE that would help a programmer to debug a program.

Explain how each of these could be used in the debugging of the `TestRandom` procedure from **part (a)**.

Feature 1

Explanation

.....

.....

.....

Feature 2

Explanation

.....

.....

.....

Feature 3

Explanation

.....

.....

.....

[6]

(c) The procedure is developed and run using the call `TestRandom(200)`. No system errors are produced.

To ensure that the procedure works correctly, you need to check the output.

Describe **two** checks you should make to suggest the program works correctly.

1

.....

.....

2

.....

.....

[2]

QUESTION 22.



- 1 (a) The following table contains statements written in pseudocode.

Show what type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Selection	Repetition (Iteration)	Assignment
WHILE Count < 20			
Count ← Count + 1			
IF MyGrade <> 'C' THEN			
Mark[Count] ← GetMark(StudentID)			
ELSE OUTPUT "Fail"			
ENDFOR			

[6]

- (b) (i) The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
MyAverage ← 13.5	
ProjectCompleted ← TRUE	
Subject ← "Home Economics"	
MyMark ← 270	
MyGrade ← 'B'	

[5]

- (ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from **part (b)(i)**.

If any expression is invalid, write "ERROR" in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
"Air-" & MID(Subject, 7, 3)	
INT(MyAverage / 2)	
ProjectCompleted AND MyMark > 270	
ProjectCompleted OR MyMark > 260	
ASC(MyGrade / 3)	

[5]

QUESTION 23.



1 (a) The following table contains statements written in pseudocode.

Show the type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Selection	Repetition (Iteration)	Assignment
Index ← Index + 5			
FOR Count ← 1 TO 100			
TempValue[Index] ← ReadValue(SensorID)			
IF Index < 30			
UNTIL DayNumber > 7			
OTHERWISE OUTPUT "ERROR"			

[6]

(b) (i) The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
Revision ← 'B'	
MaxValue ← 13.3	
ArrayFull ← TRUE	
Activity ← "Design"	
NumberOfEdits ← 270	

[5]

(ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from **part (b)(i)**.
If any expression is invalid, write "ERROR" in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
MID(Activity, 3, 4) & "ature"	
INT(MaxValue * 2)	
ArrayFull AND NumberOfEdits < 300	
ASC(Revision + 1)	
Activity = "Testing" OR Revision = 'A'	

[5]

QUESTION 24.



- 1 (a) The following table contains statements written in pseudocode.

Show the type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Assignment	Selection	Repetition (Iteration)
CASE OF TempSensor1			
ELSE			
REPEAT			
ENDFOR			
DayNumber ← DayNumber + 1			
Error ← TRUE			

[6]

- (b) (i) The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
Revision ← 500	
FuelType ← 'P'	
MinValue ← -6.3	
ServiceDue ← FALSE	
ModelRef ← "W212DEC15"	

[5]

- (ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from part (b)(i).

If any expression is invalid, write "ERROR" in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
"Month: " & MID(ModelRef, 5, 3)	
INT(MinValue * 2)	
ASC(Revision)	
Revision > 500	
ServiceDue = TRUE OR FuelType = 'P'	

[5]

QUESTION 25.



- 5 (a) Programming languages usually contain a range of built-in functions, such as a random number generator.

State **three** advantages of using built-in functions.

- 1
- 2
- 3
- [3]

- (b) A student is learning about random number generation.

She is investigating how many times the random function needs to be called before every number in a given series is generated.

She is using **pseudocode** to develop a procedure, `TestRand()`, which will:

- use the random number function to generate an integer value in the range 1 to 50 inclusive
- count how many times the random function needs to be called before all 50 values have been generated
- output a message giving the number of times the random function was called.

QUESTION 26.

4 The following is pseudocode for a string handling function.

For the built-in functions list, refer to the **Appendix** on page 16.



```
FUNCTION Search(InString : STRING) RETURNS INTEGER
```

```
    DECLARE NewString : STRING
    DECLARE Index : INTEGER
    DECLARE NextChar : CHAR
    DECLARE Selected : INTEGER
    DECLARE NewValue : INTEGER
```

```
    NewString ← '0'
    Selected ← 0
```

```
    FOR Index ← 1 TO LENGTH(InString)
```

```
        NextChar ← MID(InString, Index, 1)
        IF NextChar < '0' OR NextChar > '9'
            THEN
                NewValue ← STRING_TO_NUM(NewString)
                IF NewValue > Selected
                    THEN
                        Selected ← NewValue
                    ENDIF
                NewString ← '0'
            ELSE
                NewString ← NewString & NextChar
            ENDIF
```

```
    ENDFOR
```

```
    RETURN Selected
```

```
ENDFUNCTION
```




(b) There is an error in the algorithm. When called as shown in **part (a)(i)**, the return the largest value as expected.

(i) Explain why this error occurred when the program called the function.

.....

.....

.....

..... [2]

(ii) Describe how the algorithm could be amended to correct the error.

.....

.....

.....

QUESTION 27.



- 5 Nigel is learning about string handling. He wants to write code to count the number of words in a given string. A word is defined as a sequence of alphabetic characters that is separated by one or more space characters.

His first attempt at writing an algorithm in pseudocode is as follows:

```
PROCEDURE CountWords (Message : STRING)

    DECLARE NumWords : INTEGER
    DECLARE Index : INTEGER
    CONSTANT Space = ' '

    NumWords ← 0

    FOR Index ← 1 TO LENGTH(Message)
        IF MID(Message, Index, 1) = Space
            THEN
                NumWords ← NumWords + 1
            ENDIF
        ENDFOR

    OUTPUT "Number of words : " , NumWords

ENDPROCEDURE
```

For the built-in functions list, refer to the **Appendix** on page 18.

His first attempt is incorrect. He will use white-box testing to help him to identify the problem.

- (a) (i) State the purpose of white-box testing.

.....
..... [1]

- (ii) Dry running the code is often used in white-box testing. In this method, the programmer records the values of variables as they change.

Identify what the programmer would normally use to record the changes.

..... [1]



(b) (i) Write a test string containing two words that gives the output:

Number of words : 2

Use the symbol '▽' to represent each space character in your test string.

Explain why the algorithm gives the output shown above.

String

Explanation

.....

.....

.....

.....

[3]

(ii) Nigel tested the procedure with the strings:

String 1: "Red▽and▽Yellow"

String 2: "Green▽▽and▽▽Pink▽"

Give the output that is produced for each of the strings.

Describe the changes that would need to be made to the algorithm to give the correct output in each case.

Do **not** write pseudocode **or** program code.

String 1

Description

.....

.....

.....

String 2

Description

.....

.....

.....

[6]

QUESTION 28.



- 4 A student is developing a program to count how many times each character of the string occurs in a given string. Upper case and lower case characters will be counted as the same character. The string may contain non-alphabetic characters, which should be ignored.

The program will:

- check each character in the string to count how many times each alphabetic character occurs
- store the count for each alphabetic character in a 1D array
- output each count together with the corresponding character.

(a) The student has written a structured English description of the algorithm:

1. START at the beginning of the string
2. SELECT a character from the string
3. CONVERT the character to upper case
4. CHECK whether the character is alphabetic and INCREMENT as required.
5. REPEAT from step 2 until last character has been checked
6. OUTPUT a suitable message giving the count of each alphabetic character

Step 4 above is not described in sufficient detail.

The student decides to apply a process to increase the level of detail given in step 4.

State the name of the process **and** use this process to write step 4 in more detail. Use **structured English** for your answer.

Process

Structured English

.....

.....

.....

.....

.....

.....

[4]

QUESTION 29.



- 3 A student is developing a program to search through a string of numeric digits to count how many times each digit occurs. The variable `InString` will store the string and the 1D array `Result` will store the count values.

The program will:

- check each character in the string to count how many times each digit occurs
- record the count for each digit using the array
- output the count for each element of the array together with the corresponding digit.

- (a) The array `Result` is a 1D array of type `INTEGER`.

Write **pseudocode** to declare the array and to initialise all elements to zero.

.....

.....

.....

.....

.....

.....

.....

..... [3]

